$P \neq NP \cap \text{co-}NP$ for Infinite Time Turing Machines

Vinay Deolalikar
HEWLETT-PACKARD RESEARCH

Joel David Hamkins*
GEORGIA STATE UNIVERSITY & THE CITY UNIVERSITY OF NEW YORK

Ralf-Dieter Schindler
INSTITUT FÜR FORMALE LOGIK, UNIVERSITÄT WIEN

February 1, 2008

Abstract. Extending results of Schindler [Sch] and Hamkins and Welch [HW03], we establish in the context of infinite time Turing machines that P is properly contained in $NP \cap \text{co-}NP$. Furthermore, $NP \cap \text{co-}NP$ is exactly the class of hyperarithmetic sets. For the more general classes, we establish that $P^+ = NP^+ \cap \text{co-}NP^+ = NP \cap \text{co-}NP$, though P^{++} is properly contained in $NP^{++} \cap \text{co-}NP^{++}$. Within any contiguous block of infinite clockable ordinals, we show that $P_\alpha \neq NP_\alpha \cap \text{co-}NP_\alpha$, but if β begins a gap in the clockable ordinals, then $P_\beta = NP_\beta \cap \text{co-}NP_\beta$. Finally, we establish that $P^f \neq NP^f \cap \text{co-}NP^f$ for most functions $f : \mathbb{R} \to \text{ORD}$, although we provide examples where $P^f = NP^f \cap \text{co-}NP^f$ and $P^f \neq NP^f$.

Keywords: Infinite time Turing machines, complexity theory, descriptive set theory. Mathematics Subject Classification 03D15, 68Q15, 68Q17, 03E15.

*The research of the second author has been supported in part by grants from Georgia State University, the Research Foundation of CUNY and the National Science Foundation.

1 Introduction

In this article, we take up the question of whether $P = NP \cap \text{co-}NP$ for infinite time Turing machines. The related $P = NP$ problem was first considered in connection with infinite time Turing machines by Schindler (the third author) in [Sch], where he proved that $P \neq NP$ and introduced the other natural complexity classes P^+, NP^+, P^{++}, NP^{++}, P_α, NP_α, P^f and NP^f. He then showed that $P^+ \neq NP^+$ and posed the corresponding questions for P^{++} and P^f when f is a suitable function from $\mathbb{R}$ to the ordinals. Hamkins (the second author) and Welch answered these questions in [HW03] by showing that $P^{++} \neq NP^{++}$ and, more generally, that $P^f \neq NP^f$ for almost every function f. Here, we extend the analysis of all these complexity classes to the analogues of the question of whether $P = NP \cap \text{co-}NP$. Unfortunately, there is no uniform answer, as some of the complexity classes satisfy the equation and some do not, though the general tendency is towards inequality.

We show, in particular, that P is properly contained in $NP \cap \text{co-}NP$. Furthermore, $NP \cap \text{co-}NP$ is exactly the class of hyperarithmetic sets. At the next level, we establish $P^+ = NP^+ \cap \text{co-}NP^+ = NP \cap \text{co-}NP$. At a still higher level, once again P^{++} is properly contained in $NP^{++} \cap \text{co-}NP^{++}$. Within any contiguous block of infinite clockable ordinals, we establish $P_\alpha \neq NP_\alpha \cap \text{co-}NP_\alpha$, but if β begins a gap in the clockable ordinals, then $P_\beta = NP_\beta \cap \text{co-}NP_\beta$. Finally, for almost all functions $f : \mathbb{R} \to \textsc{ord}$, the class P^f is properly contained in $NP^f \cap \text{co-}NP^f$, though there are functions for which $P^f = NP^f \cap \text{co-}NP^f$, even with $P^f \neq NP^f$.

Infinite time Turing machines were introduced by Hamkins and Lewis in [HL00], and we refer the reader to that article for reference and background. Let us quickly describe here for convenience how the machines operate. The

			s						
input:	1	1	1	1	0	1	0	0	…
scratch:	1	0	0	1	1	0	0	1	…
output:	1	0	1	1	0	1	1	1	…

hardware of an infinite time Turing machine is the same as that of a classical three-tape Turing machine: a head moves left and right on a semi-infinite paper tape, reading and writing according to the rigid instructions of a fi-

nite program with finitely many states in exactly the classical manner. The operation of the machines is extended into transfinite ordinal time by defining the configuration of the machine at the limit ordinal stages. At such a stage, the head is returned to the leftmost cell, the machine is placed into the special *limit* state, and the tape is updated by placing into each cell the lim sup of the values appearing in that cell before the limit stage. Thus, if the cell values have stabilized before a limit, then at the limit the cell displays this stabilized value, and otherwise, when the cell has changed from 0 to 1 and back again unboundedly often before the limit, then at the limit the cell displays a 1. Having specified the operation of the machines, one obtains for any program p the corresponding infinite time computable function φ_p, namely, $\varphi_p(x) = y$ when program p on input x halts with output y. The natural context for input and output is infinite binary sequences, that is, Cantor space ${}^\omega 2$, which we refer to as the set of reals and denote by $\mathbb{R}$. A set $A \subseteq \mathbb{R}$ is infinite time decidable if its characteristic function is decidable. In the context of certain time-critical complexity classes, we adopt the formalism for deciding sets with two distinct halt states, *accept* and *reject*, so that the machines can announce their decisions as quickly as possible, without needing to position the head for writing on the output tape. For many of the complexity classes, however, including P, P^+, P^{++} and P_α for limit ordinals α and their successors, the additional steps required for writing on the tape pose no difficulty, and one can dispense with this formalism in favor of the usual characteristic function notion of decidability.

Many of our arguments will rely on elementary results in descriptive set theory, and we refer readers to [Mos80], [Kec95] and [MW85] for excellent introductions. For background material on admissible set theory, we refer readers to [Bar75]. We denote the first infinite ordinal by ω and the first uncountable ordinal by ω_1. Throughout the paper, we use ordinal as opposed to cardinal arithmetic in such expressions as ω^2 and ω^ω. The well-known ordinal ω_1^{ck}, named for Church and Kleene, is the supremum of the recursive ordinals (those that are the order type of a recursive relation on ω). The ordinal ω_1^{ck} is also the least *admissible* ordinal, meaning that the ω_1^{ck} level of Gödel's constructible universe $L_{\omega_1^{\mathrm{ck}}}$ satisfies the Kripke-Platek (KP) axioms of set theory. We denote by ω_1^x the supremum of the x-recursive ordinals, and this is the same as the least x-admissible ordinal, meaning that $L_{\omega_1^x}[x] \models KP$. An ordinal α is *clockable* if there is a computation of the form $\varphi_p(0)$ taking exactly α many steps to halt (meaning that the α^{th} step moves into the *halt* state). A *writable* real is one that is the output of a computation $\varphi_p(0)$. An

ordinal is writable when it is coded by a writable real. The supremum of the writable ordinals is denoted λ, and by [Wel00] this is equal to the supremum of the clockable ordinals. A real is *accidentally writable* when it appears on one of the tapes at same stage during a computation of the form $\varphi_p(0)$. The supremum of the accidentally writable ordinals, those that are coded by an accidentally writable real, is denoted Σ. A real is *eventually writable* if there is a computation of the form $\varphi_p(0)$ such that beyond some ordinal stage the real is written on the output tape (the computation need not halt). Ordinals coded by such reals are also said to be eventually writable, and we denote the supremum of the eventually writable ordinals by ζ. Results in [HL00] establish that $\lambda < \zeta < \Sigma$ and that λ and ζ are admissible. Welch [Wel00] established that every computation $\varphi_p(0)$ either halts before λ or else repeats the ζ configuration at Σ, in a transfinitely repeating loop. Furthermore, these ordinals are optimal in the sense that the universal computation that simulates all $\varphi_p(0)$ simultaneously first enters its repeating loop at ζ, first repeating it at Σ. It follows that Σ is not admissible.

The research in this article was initiated by the first author in a preliminary paper, which was subsequently refined and expanded into the current three-author collaboration.

2 Defining the Complexity Classes

Let us quickly recall the definitions of the complexity classes.

Schindler [Sch] generalized the class of polynomial decidable sets to the infinite time context with the natural observation that every input $x \in \mathbb{R}$ has length ω, and so the sets in P should be those that are decidable in fewer steps than a polynomial function of ω. Since all such polynomials are bounded by those of the form ω^n for $n \in \omega$, he defined for $A \subseteq \mathbb{R}$ that $A \in P$ when there is a infinite time Turing machine T and a natural number n such that T decides A and T halts on every input in fewer than ω^n many steps.

The corresponding nondeterministic class was defined by $A \in NP$ if there is an infinite time Turing machine T and a natural number n such that $x \in A$ if and only if there is $y \in \mathbb{R}$ such that T accepts (x, y), and T halts on every input in fewer than ω^n many steps. Sets in NP are therefore simply the projections of sets in P.

The class P occupies a floor a little ways upwards in the skyscraper hierarchy of classes P_α, indexed by the ordinals, where $A \in P_\alpha$ if and only if

there is a Turing machine T and an ordinal $\beta < \alpha$ such that T decides A, and T halts on every input in fewer than β many steps. In this notation, the polynomial class P is simply P_{ω^ω}, while the hierarchy continues up through the countable ordinals to P_{ω_1}, the class of sets that are decidable uniformly by some countable stage, and P_{ω_1+1}, the class of all decidable sets. We admit that the term "polynomial" and the letter P are perhaps only appropriate at the level of P_{ω^ω}, as one might naturally view P_{ω^2} instead as the "linear time" sets, $P_{\omega^{\omega^2}}$ as the "exponential time" sets, P_{ϵ_0} as the "super-exponential time" sets, $P_{\omega_1^{\mathrm{ck}}}$ as the "computable time" sets, and so on, though at some point (probably already well exceeded) such analogies become strained. Nevertheless, we retain the symbol P in P_α as suggesting the polynomial time context of classical complexity theory, because we have placed limitations on the lengths of allowed computations. After all, infinite time Turing machines can profitably use computations of any countable length, and so any uniform restriction to a particular countable α is a severe limitation. Since all these classes concern infinite computations, one should not regard them as feasible in any practical sense.

One defines the nondeterministic hierarchy in a similar manner: $A \in NP_\alpha$ if there is a Turing machine T and $\beta < \alpha$ such that $x \in A$ if and only if there is $y \in \mathbb{R}$ such that T accepts (x, y), and T halts on every input in fewer than β steps. In this notation, NP is NP_{ω^ω}. Clearly, the sets in NP_α are simply the projections of sets in P_α.

As is usual in the classical context, for the nondeterministic classes we assume that the witness y is provided on a separate input tape, rather than coded together with x on one input tape. This is necessary because in order to know $P_\alpha \subseteq NP_\alpha \cap \text{co-}NP_\alpha$, one wants to be able to ignore the witness y without needing extra steps of computation. When α is a limit ordinal or the successor of a limit ordinal, however, one can easily manage without an extra input tape, because there is plenty of time to decode both x and the verifying witness y from one input tape.

So far, these complexity classes treat every input equally in that they impose uniform bounds on the lengths of computation, independently of the input. But it may seem more natural to allow a more complicated input to have a longer computation. For this reason, taking ω_1^x as a natural measure of the complexity of x, Schindler defined $A \in P^+$ when there is an infinite time Turing machine deciding A and halting on input x in fewer than ω_1^x many steps. The corresponding nondeterministic class is defined by $A \in NP^+$ when there is an infinite time Turing machine T such that $x \in A$ if and only if

there is $y \in \mathbb{R}$ such that T accepts (x, y), and T halts on input (x, y) in fewer than ω_1^x many steps. Because this bound depends only on x and not on y, one can't conclude immediately that NP^+ is the projection of P^+. One of the surprising results of the analysis, however, is that the apparent extra power of allowing computations on input x to go up to ω_1^x, as opposed to merely ω_1^{ck}, actually provides no advantage (see the discussion following Theorem 5). Consequently, NP^+ is the projection of P^+ after all.

Allowing computations to proceed a bit longer, Schindler defined that $A \in P^{++}$ when there is an infinite time Turing machine deciding A and halting on input x in at most $\omega_1^x + \omega$ many steps. Similarly, $A \in NP^{++}$ when there is an infinite time Turing machine T such that $x \in A$ if and only if there is $y \in \mathbb{R}$ such that T accepts (x, y), and T halts on any input (x, y) in at most $\omega_1^x + \omega$ many steps.

Finally, Schindler observed that any function f from $\mathbb{R}$ to the ordinals can be viewed as bounding a complexity class, namely, $A \in P^f$ if there is an infinite time Turing machine deciding each $x \in A$ in fewer than $f(x)$ many steps.[1] And $A \in NP^f$ when there is an infinite time Turing machine T such that $x \in A$ if and only if there is $y \in \mathbb{R}$ such that T accepts (x, y), and T halts on any input (x, y) in fewer than $f(x)$ many steps. In this notation, P^+ is the class P^{f_0}, where $f_0(x) = \omega_1^x + 1$, and $P^{++} = P^{f_1}$, where $f_1(x) = \omega_1^x + \omega + 1$.

3 Proving $P \neq NP \cap \text{co-}NP$

We begin with the basic result separating P from $NP \cap \text{co-}NP$. In subsequent results we will improve on this and precisely characterize the set $NP \cap \text{co-}NP$.

Theorem 1 *$P \neq NP \cap \text{co-}NP$ for infinite time Turing machines.*

Proof: Clearly P is contained in NP and closed under complements, so it follows that $P \subseteq NP \cap \text{co-}NP$. We now show that the inclusion is proper. Consider the halting problem for computations halting before ω^ω given by

$$h_{\omega^\omega} = \{\, p \mid \varphi_p(p) \text{ halts in fewer than } \omega^\omega \text{ steps} \,\}.$$

We claim that $h_{\omega^\omega} \notin P$. This follows from [HL00, Theorem 4.4] and is an instance of Lemma 8 later in this article, but let us quickly give the

[1] This definition differs from that in [HW03], which allows $\leq f(x)$ many steps in order to avoid the inevitable $+1$ that occurs when defining such classes as P^+ and P^{++}. Here we use the original definition of [Sch], which is capable of describing more classes.

argument. If we could decide h_{ω^ω} in time before ω^ω, then we could compute the function $f(p) = 1$, if $p \notin h_{\omega^\omega}$, diverge otherwise, and furthermore we could compute this function in time before ω^ω for input $p \notin h_{\omega^\omega}$. If this algorithm for computing f is carried out by program q, then $q \notin h_{\omega^\omega}$ if and only if $f(q) \downarrow = 1$, which holds if and only if $\varphi_q(q)$ halts in fewer than ω^ω steps, which holds if and only if $q \in h_{\omega^\omega}$, a contradiction.

Let us now show that $h_{\omega^\omega} \in NP$. The idea of the proof is that the question of whether $p \in h_{\omega^\omega}$ can be verified by inspecting (a code for) the computation sequence of $\varphi_p(p)$ up to ω^ω. Specifically, to set this up, fix a recursive relation $\lhd$ on ω having order type ω^ω and a canonical computable method of coding infinite sequences of reals as reals, so that we may interpret any real z as an infinite sequence of reals $\langle z_n \mid n \in \omega \rangle$. By combining this coding with the relation $\lhd$, we may view the index n as representing the ordinal α of its order type with respect to $\lhd$, and we have a way to view any real z as an ω^ω-sequence of reals $\langle (z)_\alpha \mid \alpha < \omega^\omega \rangle$. This coding is computable in the sense that given any $n \in \omega$ representing α with respect to $\lhd$, we can uniformly compute any digit of $(z)_\alpha$.

Now consider the algorithm accepting input (p, z) exactly when with respect to the above coding the real z codes a halting sequence of snapshots $\langle (z)_\alpha \mid \alpha < \omega^\omega \rangle$ of the computation $\varphi_p(p)$. That is, first, each $(z)_\alpha$ codes the complete configuration of an infinite time Turing machine, including the contents of the tapes, the position of the head, the state and the program; second, the snapshot $(z)_{\alpha+1}$ is computed correctly from the previous snapshot $(z)_\alpha$, taking the convention that the snapshots should simply repeat after a halt; third, the limit snapshots $(z)_\lambda$ for limit ordinals λ are updated correctly from the previous snapshots $(z)_\alpha$ for $\alpha < \lambda$; and finally, fourth, one of the snapshots shows the computation to have halted. Since all of these requirements form ultimately merely an arithmetic condition on the code z, they can be checked by an infinite time Turing machine in time uniformly before ω^2. And since $p \in h_{\omega^\omega}$ if and only if the computation sequence for $\varphi_p(p)$ halts before ω^ω, we conclude that $p \in h_{\omega^\omega}$ exactly if there is a real z such that (p, z) is accepted by this algorithm. Thus, $h_{\omega^\omega} \in NP$.

To see that $h_{\omega^\omega} \in$ co-NP, we simply change the fourth requirement to check that none of the snapshots show the computation to have halted. This change means that the input (p, z) will be accepted exactly when z codes a sequence of snapshots of the computation $\varphi_p(p)$, exhibiting it not to have halted in ω^ω many steps. Since there is a real z like this if and only if $p \notin h_{\omega^\omega}$, it follows that the complement of h_{ω^ω} is in NP, and so $h_{\omega^\omega} \in$ co-NP. $\square$

Because the verification algorithm needed only to check an arithmetic condition, the argument actually establishes $h_{\omega^\omega} \in NP_{\omega^2} \cap \text{co-}NP_{\omega^2}$. A closer analysis reveals that the requirements that need to be checked are Π^0_3 (one must check that every code for a cell at a limit stage has the right value). And since any Π^0_3 statement can be decided in time $\omega + \omega$, it follows that h_{ω^ω} is in $NP_{\omega\cdot 2+2} \cap \text{co-}NP_{\omega\cdot 2+2}$. In fact, a bit of thought shows that the verification idea of the proof shows that any set in *NP* can be verified by inspecting a snapshot sequence of length ω^ω, so we may actually conclude $NP = NP_{\omega\cdot 2+2}$ and $\text{co-}NP = \text{co-}NP_{\omega\cdot 2+2}$. We now push these ideas harder, down to the (optimal) level of $\omega + 2$, by asking more of our witnesses.

Theorem 2 *The classes NP_α for $\omega + 2 \leq \alpha \leq \omega_1^{\text{ck}}$ are all identical to the class Σ^1_1 of lightface analytic sets. In particular, $NP = NP_{\omega+2}$, and so membership in any NP set can be verified in only ω many steps. Similarly, the corresponding classes* co-NP_α *are all identical to the Π^1_1 sets. Consequently, $NP \cap \text{co-}NP$ is exactly the class Δ^1_1 of hyperarithmetic sets.*

Proof: The idea is to have a witness not merely of the computation sequence of a given computation, but also of all arithmetic truths. To recognize the validity of such witnesses in ω many steps, we make use of the following two lemmas.

Lemma 2.1 *Any $\Pi^0_2(x)$ statement can be decided on input x in ω many steps.*

Proof: To decide the truth of $\forall n \exists m \psi(n, m, x)$, where ψ has only bounded integer quantifiers, one systematically considers each n in turn, searching for a witness m that works with that n. Each time this succeeds, move to the next n and flash a master flag on and then off again. If the flag is on at a limit, it means that infinitely many n were considered, so the statement is true. If the flag is off, it means that for some n the search for a witness m was never completed, so the statement is false. $\square$

Lemma 2.2 *There is an infinite time Turing machine algorithm deciding in ω many steps on input (a, A) whether A is the set of arithmetic truths in a.*

Proof: It is easy to see by induction on formulas that $A \subseteq \omega$ is the set of codes for true arithmetic statements in a (that is, using $a \subseteq \omega$ as a predicate in the language) if and only if the following conditions, using a recursive Gödel coding $\ulcorner\psi\urcorner$, are satisfied:

(*i*) If ψ is atomic, then $\ulcorner\psi\urcorner \in A$ if and only if ψ is true.

(*ii*) $\ulcorner\neg\psi\urcorner \in A$ if and only if $\ulcorner\psi\urcorner \notin A$.

(*iii*) $\ulcorner\psi \wedge \phi\urcorner \in A$ if and only if $\ulcorner\psi\urcorner \in A$ and $\ulcorner\phi\urcorner \in A$.

(*iv*) $\ulcorner\exists u\psi(u)\urcorner \in A$ if and only if there is a natural number n such that $\ulcorner\psi(n)\urcorner \in A$.

The first three of these conditions are primitive recursive in (a, A), while the fourth has complexity Π^0_2 in (a, A), making the overall complexity Π^0_2 in (a, A). It follows from Lemma 2.1 that whether or not (a, A) satisfies these four conditions can be checked in ω many steps. More concretely, we can describe an algorithm: we systematically check that A satisfies each of the conditions by considering each Gödel code in turn. For a fixed formula, the first three conditions can be checked in finite time. For the fourth condition, given a code for $\psi(n)$ in A, the algorithm can check whether the code for $\exists u\psi(u)$ is in A; conversely, given that $\exists u\psi(u)$ is in A, let the algorithm search for an n such that $\psi(n)$ is in A. The point, as in Lemma 2.1, is that if this search fails, then at the limit one can reject the input without more ado, since it has failed Condition (iv). Otherwise, a witness n is found in finitely many steps, and the next formula is considered. $\square$

Returning to the proof of Theorem 2, we now prove that when $\omega + 2 \leq \alpha \leq \omega_1^{\mathrm{ck}}$, the classes NP_α are identical. Since this is clearly a nondecreasing sequence of classes, it suffices to show $NP_{\omega_1^{\mathrm{ck}}} \subseteq NP_{\omega+2}$. For this, consider any set $B \in NP_{\omega_1^{\mathrm{ck}}}$. By definition, this means that there is a program p and a recursive ordinal β such that $\varphi_p(x, y)$ halts in time β for all input and $x \in B$ if and only if there is a y such that φ_p accepts (x, y). Fix a recursive relation on ω having order type β. Consider the algorithm that accepts input (x, y, z, A) exactly when A codes the set of arithmetic truths in (x, y, z) and z codes the computation sequence of $\varphi_p(x, y)$ of length β (using the fixed recursive relation for β as the underlying order of the snapshots), and this computation sequence shows the computation to have accepted the input. We claim that this algorithm halts in just ω many steps. To see this, observe first that the latter part of the condition, about z coding the computation sequence for $\varphi_p(x, y)$, is arithmetic in (x, y, z). Therefore, by trusting momentarily that A is correct, it can be verified in finitely many steps by simply checking whether the Gödel code of that arithmetic condition is in A. After this, one can verify

in ω many steps that A is in fact correct using the algorithm of Lemma 2.2. So altogether we can decide whether (x, y, z, A) has these properties in just ω many steps. And since $x \in B$ if and only if φ_p accepts (x, y), and this happens just in case (x, y, z, A) is accepted by our algorithm, where z codes the computation sequence of length β and A codes the arithmetic truths in (x, y, z), we conclude that $B \in NP_{\omega+2}$, as desired. We have therefore proved $NP_{\omega_1^{\mathrm{ck}}} \subseteq NP_{\omega+2}$, and so the classes NP_α are identical for $\omega + 2 \leq \alpha \leq \omega_1^{\mathrm{ck}}$.

We now draw the remaining conclusions stated in the theorem. Since NP simply denotes NP_{ω^ω}, falling right in the middle of the range, it follows that $NP = NP_{\omega+2}$, and so membership in any NP set can be verified in ω many steps. By [HL00, Theorem 2.7] we know that $P_{\omega_1^{\mathrm{ck}}} = \Delta^1_1$. It follows immediately that $NP_{\omega_1^{\mathrm{ck}}} = \Sigma^1_1$, as these sets are the projections of sets in $P_{\omega_1^{\mathrm{ck}}}$. So $NP_\alpha = \Sigma^1_1$ whenever $\omega + 2 \leq \alpha \leq \omega_1^{\mathrm{ck}}$, as these classes are all identical. And finally, by taking complements, we conclude as well that co-$NP_\alpha = \Pi^1_1$ whenever $\omega + 2 \leq \alpha \leq \omega_1^{\mathrm{ck}}$. $\Box$

It will follow from Theorem 5 that this result can be extended at least one more step, to $\omega_1^{\mathrm{ck}} + 1$, because $NP_{\omega_1^{\mathrm{ck}}} = NP_{\omega_1^{\mathrm{ck}}+1}$.

Corollary 3 *$NP \neq$ co-NP for infinite time Turing machines.*

Proof: The classes Σ^1_1 and Π^1_1 are not identical. $\Box$

Both Theorems 1 and 2 can also be proved using the model-checking technique of [HW03], which we will use extensively later in this article.

4 Proving $P^+ = NP^+ \cap \text{co-}NP^+$

At first glance, the class P^+ appears much more generous than the earlier classes, because computations on input x are now allowed up to ω_1^x many steps, which can be considerably larger than ω_1^{ck}. But it will follow from Theorem 5 that if a set is in P^+, then there is an algorithm deciding it in uniformly fewer than ω_1^{ck} many steps, much sooner than required. Our arguments rely on the following fact from descriptive set theory.

Lemma 4 *Π^1_1 absorbs existential quantification over Δ^1_1. That is, if B is Π^1_1 and $x \in A \iff \exists y \in \Delta^1_1(x)\, B(x, y)$, then A is Π^1_1 as well.*

Proof: This lemma is a special case of [Mos80, Theorem 4D.3], and is due to Kleene. We provide a proof sketch here. Let U be a universal Π^1_1 set and suppose $y \in \Delta^1_1(x)$. Then there is an integer i_0 such that $y(n) = m$ if and only if $U(i_0, x, n, m)$. Let U^* be a Π^1_1 set uniformizing U, so that for all i, x, n if there is an m with $U(i, x, n, m)$ then there is a unique m with $U^*(i, x, n, m)$. In particular, $y(n) = m$ if and only if $U^*(i_0, x, n, m)$. So we have altogether that $x \in A$ if and only if there is an integer i such that for all n there is exactly one m with $U^*(i, x, n, m)$, and for all y, either $B(x, y)$ or there are n, m with $U^*(i, x, n, m)$ and $y(n) \neq m$. In other words, we say that there is an index i of a computation of a real z via a $\Pi^1_1(x)$ recursive function such that $B(x, z)$. As Π^1_1 is closed under quantification over integers, this shows that A is in Π^1_1, as desired. $\square$

Theorem 5

(i) $NP^+ = \Sigma^1_1 = NP = NP_\alpha$ *whenever* $\omega + 2 \leq \alpha \leq \omega_1^{\mathrm{ck}} + 1$.

(ii) $P^+ = \Delta^1_1 = P_{\omega_1^{\mathrm{ck}}} = P_{\omega_1^{\mathrm{ck}}+1}$.

(iii) $P^+ = NP^+ \cap \text{co-}NP^+$.

Proof: For (i), we have already proved in Theorem 2 that $\Sigma^1_1 = NP$, and since clearly $NP \subseteq NP^+$, it follows that $\Sigma^1_1 \subseteq NP^+$. Conversely, suppose that $A \in NP^+$. This means that there is an infinite time Turing machine program p such that $\varphi_p(x, y)$ halts on all input (x, y) in fewer than ω_1^x many steps, and $x \in A$ if and only if there is a real y such that φ_p accepts (x, y). The set A is therefore the projection of the set

$$B = \{\, (x, y) \mid \varphi_p \text{ accepts } (x, y) \,\}.$$

In order to see that A is in Σ^1_1, it suffices to show $B \in \Sigma^1_1$ (and our argument shows just as easily that $B \in \Delta^1_1$). The complement of B is the set $\neg B = \{\, (x, y) \mid \varphi_p \text{ rejects } (x, y) \,\}$, and these computations also have length less than ω_1^x. It follows that the computation sequence for $\varphi_p(x, y)$ exists in the model $L_{\omega_1^x}[x, y]$, and so $(x, y) \in \neg B$ if and only if $L_{\omega_1^x}[x, y] \models \theta(x, y)$, where $\theta(x, y)$ asserts that the computation $\varphi_p(x, y)$ rejects the input. Since this is a Σ_1 assertion, it follows that $(x, y) \in \neg B$ if and only if there is an ordinal $\beta < \omega_1^x$ such that $L_\beta[x, y] \models \theta(x, y)$. Since the model $L_\beta[x, y]$ is hyperarithmetic

in (x, y), and any well-founded model showing the computation to reject the input will do, we see that $(x, y) \in \neg B$ if and only if there is a real $z \in \Delta^1_1(x, y)$ coding a well-founded model of $V = L[x, y]$ that satisfies $\theta(x, y)$. Since the property of coding a well-founded model (of any theory) is Π^1_1 in the theory, it follows by Lemma 4 that $\neg B$ is Π^1_1. Consequently, $B \in \Sigma^1_1$, and so A, being the projection of B, is in Σ^1_1 as well. So we have proved that $NP^+ = \Sigma^1_1$. It follows from Theorem 2 that $NP^+ = NP_\alpha$ whenever $\omega + 2 \leq \alpha \leq \omega_1^{\mathrm{ck}}$. The remaining case of $\alpha = \omega_1^{\mathrm{ck}} + 1$ follows from (ii) and the observation that $NP_{\omega_1^{\mathrm{ck}}} = NP_{\omega_1^{\mathrm{ck}}+1}$, as these are the projections of $P_{\omega_1^{\mathrm{ck}}} = P_{\omega_1^{\mathrm{ck}}+1}$.

For (ii) and (iii), observe that since $NP^+ = \Sigma^1_1$, it follows that co-$NP^+ = \Pi^1_1$, and so $P^+ \subseteq NP^+ \cap \text{co-}NP^+ = \Sigma^1_1 \cap \Pi^1_1 = \Delta^1_1$, which by [HL00, Theorem 2.7] is equal to $P_{\omega_1^{\mathrm{ck}}}$, which is a subset of $P_{\omega_1^{\mathrm{ck}}+1}$, which is clearly a subset of P^+. So all of them are equal, as we claimed. $\square$

The fact that $P^+ = \Delta^1_1$ was Theorem 2.13 of [Sch], and one can view our argument here as a detailed expansion of that argument. In fact, however, once one knows $P^+ = P_{\omega_1^{\mathrm{ck}}} = \Delta^1_1$, it follows immediately that sets in NP^+ are projections of sets in $P_{\omega_1^{\mathrm{ck}}} = \Delta^1_1$, since all the computations halt uniformly before ω_1^{ck}, which is certainly not larger than ω_1^x, and consequently $NP^+ = \Sigma^1_1$. By this means, Theorem 5 follows directly from [Sch, Theorem 2.13].

The fact that $P^+ = P_{\omega_1^{\mathrm{ck}}}$ should be surprising—and we mentioned this in the introduction—because it means that although the computations deciding $x \in A$ for $A \in P^+$ are allowed to compute up to ω_1^x, in fact there is an algorithm needing uniformly fewer than ω_1^{ck} many steps. So the difference between ω_1^{ck} and ω_1^x, which can be substantial, gives no advantage at all in computation. An affirmative answer to the following question would explain this phenomenon completely.

Question 6 *Suppose an algorithm halts on each input x in fewer than ω_1^x steps. Then does it halt uniformly before ω_1^{ck}?*

Secondly, the fact that $P_{\omega_1^{\mathrm{ck}}} = P_{\omega_1^{\mathrm{ck}}+1}$ is itself surprising, because the difference in the definitions of these two classes is exactly the difference between requiring the computations to halt before ω_1^{ck} and requiring them to halt *uniformly* before ω_1^{ck}, that is, before some fixed $\beta < \omega_1^{\mathrm{ck}}$ on all input. Since the classes $P_{\omega_1^{\mathrm{ck}}} = P_{\omega_1^{\mathrm{ck}}+1}$ are equal, any set that can be decided before ω_1^{ck} can be decided uniformly before ω_1^{ck}.

Finally, let us close this section with a more abstract view of Theorem 5. Suppose that $f : \mathbb{R} \to \omega_1$ is Turing invariant and for some Σ_1 formula φ we have $f(x) = \alpha$ if and only if $L[x] \models \varphi(x, \alpha)$. We define the pointclass Γ^f by $A \in \Gamma^f$ if and only if there is some Σ_1 formula θ such that $x \in A \iff L_{f(x)}[x] \models \theta(x)$. Then for "natural" f one should be able to show that $P^f = \Delta^f = NP^f \cap \text{co-}NP^f$ and $NP^f = \Gamma^f$-dual, by our arguments above. (We do not attempt to classify the functions f for which these equations hold true.) For $f(x) = \omega_1^x$, these equations collapse to the the statement of Theorem 5. The pointclasses Γ^f exhaust all of Δ^1_2.

5 The Question Whether $P_\alpha = NP_\alpha \cap \text{co-}NP_\alpha$

We turn now to the relation between P_α and $NP_\alpha \cap \text{co-}NP_\alpha$ for various ordinals α. We begin with the observation that the classes P_α increase with every clockable limit ordinal α.

Definition 7 The lightface *halting problem* is the set $h = \{\, p \mid \varphi_p(p) \text{ halts}\}$. Approximating this, for any ordinal α the *halting problem for* α is the set $h_\alpha = \{\, p \mid \varphi_p(p) \text{ halts in fewer than } \alpha \text{ many steps}\}$. We sometimes denote $h_{\alpha+1}$ by $h_{\leq\alpha}$, to emphasize the fact that it is concerned with computations of length less than or equal to α. Similarly, we denote $P_{\alpha+2}$, $NP_{\alpha+2}$ and $\text{co-}NP_{\alpha+2}$ by $P_{\leq\alpha}$, $NP_{\leq\alpha}$ and $\text{co-}NP_{\leq\alpha}$, respectively, as these classes are also concerned only with the computations of length less than or equal to α. It follows that $NP_{\leq\alpha}$ is the projection of $P_{\leq\alpha}$, and $\text{co-}NP_{\leq\alpha}$ consists of the complements of sets in $NP_{\leq\alpha}$.

Lemma 8 *If α is any ordinal, then $h_\alpha \notin P_\alpha$. Indeed, $h_\alpha \notin P_{\alpha+1}$. In particular, $h_{\leq\alpha} \notin P_{\leq\alpha}$. However, if α is a clockable limit ordinal, then $h_\alpha \in P_{\leq\alpha}$.*

Proof: Suppose to the contrary that $h_\alpha \in P_{\alpha+1}$ for some ordinal α. It follows that there is an algorithm q deciding h_α in fewer than α many steps. That is, the computation of $\varphi_q(x)$ halts in fewer than α many steps on any input and $x \in h_\alpha$ if and only if φ_q accepts x. Consider the modified algorithm q_0 that runs $\varphi_q(x)$, but when the algorithm is just about to move into the *accept* state, it instead jumps into a non-halting transfinite repeating loop. This algorithm computes a function $\varphi_{q_0}(x)$ which halts in fewer than α steps if $x \notin h_\alpha$ and diverges otherwise. Therefore, $q_0 \in h_\alpha$ if and only if $\varphi_{q_0}(q_0)$ halts, which holds if and only if $q_0 \notin h_\alpha$, a contradiction. So we

have established $h_\alpha \notin P_{\alpha+1}$ for any ordinal α. It follows, in particular, that $h_{\leq\alpha} = h_{\alpha+1} \notin P_{\alpha+2} = P_{\leq\alpha}$.

Finally, when α is a clockable limit ordinal, consider the algorithm that on input p simulates both the computation $\varphi_p(p)$ and the α clock (simulating ω many steps of each in every ω many actual steps). If the computation stops before the clock runs out, the algorithm accepts the input, but if the clock runs out, it rejects the input. By placing the first column of the clock's computation in the actual first column, the algorithm will be able to detect that the clock has stopped at exactly stage α, and thereby halt in α steps. So $h_\alpha \in P_{\leq\alpha}$. $\square$

Corollary 9 *If α is a clockable limit ordinal, then $P_\alpha \subsetneq P_{\leq\alpha}$.*

For recursive ordinals α and even $\alpha \leq \omega_1^{\mathrm{ck}} + 1$, the question whether $P_\alpha = NP_\alpha \cap \text{co-}NP_\alpha$ is already settled by Theorem 2, and we summarize the situation here.

Theorem 10 *$P_\alpha \neq NP_\alpha \cap \text{co-}NP_\alpha$ whenever $\omega + 2 \leq \alpha < \omega_1^{\mathrm{ck}}$. Equality is attained at ω_1^{ck} and its successor with*

$$P_{\omega_1^{\mathrm{ck}}} = NP_{\omega_1^{\mathrm{ck}}} \cap \text{co-}NP_{\omega_1^{\mathrm{ck}}} = \Delta_1^1 = P_{\omega_1^{\mathrm{ck}}+1} = NP_{\omega_1^{\mathrm{ck}}+1} \cap \text{co-}NP_{\omega_1^{\mathrm{ck}}+1}\,.$$

Proof: For $\alpha < \omega_1^{\mathrm{ck}}$ we know by Corollary 9 that P_α is a proper subset of $P_{\omega_1^{\mathrm{ck}}}$, which by Theorem 2 is equal to $NP_{\omega_1^{\mathrm{ck}}} \cap \text{co-}NP_{\omega_1^{\mathrm{ck}}} = NP_\alpha \cap \text{co-}NP_\alpha$. So none of the earlier classes P_α are equal to $NP_\alpha \cap \text{co-}NP_\alpha$; but at the top we do achieve the equalities $P_{\omega_1^{\mathrm{ck}}} = NP_{\omega_1^{\mathrm{ck}}} \cap \text{co-}NP_{\omega_1^{\mathrm{ck}}}$ and $P_{\omega_1^{\mathrm{ck}}+1} = NP_{\omega_1^{\mathrm{ck}}+1} \cap \text{co-}NP_{\omega_1^{\mathrm{ck}}+1}$ because by Theorem 5 these are both instances of the identity $\Delta_1^1 = \Sigma_1^1 \cap \Pi_1^1$. $\square$

Corollary 9 and Theorem 10 show that the class Δ_1^1 of hyperarithmetic sets is ramified by the increasing hierarchy $\cup_{\alpha<\omega_1^{\mathrm{ck}}} P_\alpha$ in a way similar to the traditional hyperarithmetic hierarchy $\Delta_1^1 = \cup_{\alpha<\omega_1^{\mathrm{ck}}} \Delta_\alpha^0$, and one can probably give a tight analysis of the interaction of these two hierarchies.

We now prove that the pattern of Theorems 2 and 10—where the classes NP_α are identical for α in the range from $\omega + 2$ up to $\omega_1^{\mathrm{ck}} + 1$—is mirrored higher up, within any contiguous block of clockable ordinals. It will follow that P_α is properly contained in $NP_\alpha \cap \text{co-}NP_\alpha$ within any such block of clockable ordinals. We subsequently continue the pattern at the top of any such block, by proving that $P_\beta = NP_\beta \cap \text{co-}NP_\beta$ for the ordinal β that begins the next gap in the clockable ordinals.

Theorem 11 *If $[\nu, \beta)$ is a contiguous block of infinite clockable ordinals, then all the classes NP_α for $\nu + 2 \leq \alpha \leq \beta + 1$ are identical. Consequently, all the corresponding classes* co-NP_α *for such α are identical as well.*

Proof: Since the sequence of classes NP_α is nondecreasing, it suffices to show $NP_{\beta+1} \subseteq NP_{\nu+2}$. Suppose $B \in NP_{\beta+1}$, so that there is an algorithm e such that $\varphi_e(x, y)$ halts on every input in in time less than β, and $x \in B$ if and only if there is y such that φ_e accepts (x, y). Since ν is clockable, there is a program q_0 such that $\varphi_{q_0}(0)$ takes exactly ν steps to halt.

Consider the algorithm which on input (x, z) checks, first, whether z codes a model M_z of KP containing x in which the computation $\varphi_{q_0}(0)$ halts and there is a $y \in M_z$ such that φ_e accepts (x, y); and second, verifies that M_z is well-founded up to ν^{M_z}, the length of the clock computation $\varphi_{q_0}(0)$ in M_z. If both of these requirements are satisfied, then the algorithm accepts the input, and otherwise rejects it.

If $x \in B$, then there is a y such that φ_e accepts (x, y), and so we may choose z coding a fully well-founded model M_z that is tall enough to see this computation and $\varphi_{q_0}(0)$. It follows that (x, z) will be accepted by our algorithm. Conversely, if (x, z) is accepted by our algorithm, then the corresponding model M_z is well-founded up to ν. Since the well-founded part is admissible and no clockable ordinal is admissible, M_z must be well-founded beyond the length of the computation $\varphi_e(x, y)$ (which is less than β), since all the ordinals in $[\nu, \beta)$ are clockable. Therefore, M_z will have the correct (accepting) computation for $\varphi_e(x, y)$, and so $x \in B$. Thus, our algorithm nondeterministically decides B. And as before, since ν is inadmissible, this algorithm will either discover ill-foundedness below ν, halting in time at most ν, or else halt at ν with well-foundedness up to $\nu^{M_z} = \nu$. So $B \in NP_{\nu+2}$, as desired.

The corresponding fact for co-NP_α follows by taking complements. $\square$

Corollary 12 *$P_\alpha \neq NP_\alpha \cap$ co-NP_α for any clockable ordinal $\alpha \geq \omega + 2$, except possibly when α ends a gap in the clockable ordinals or is the successor of such a gap-ending ordinal.*

Proof: If $\alpha \geq \omega + 2$ is clockable but is neither a gap-ending ordinal nor the successor of a gap-ending ordinal, then there is an infinite ordinal $\nu < \nu + 2 \leq \alpha$ such that $[\nu, \alpha]$ is a contiguous block of clockable ordinals. By

Theorem 11, the classes NP_ξ are identical for $\nu + 2 \leq \xi \leq \beta + 1$, where β is the next admissible beyond α. Since by Corollary 9 the corresponding classes P_ξ increase at every clockable limit ordinal in this range and are subsets of $NP_\xi \cap \text{co-}NP_\xi$, it follows that $P_\alpha \subsetneq NP_\alpha \cap \text{co-}NP_\alpha$. $\square$

Corollary 13 *In particular, $P_{\leq\alpha} \neq NP_{\leq\alpha} \cap \text{co-}NP_{\leq\alpha}$ for any infinite clockable ordinal α.*

Proof: This is an instance of the previous theorem, because $P_{\leq\alpha} = P_{\alpha+2}$ and $\alpha + 2$ is neither a limit ordinal nor the successor of a limit ordinal. $\square$

Because of the possible exceptions in Corollary 12 at the gap-ending ordinals, we do not have a complete answer to the following question.

Question 14 *Is $P_\alpha \neq NP_\alpha \cap \text{co-}NP_\alpha$ for any clockable ordinal $\alpha \geq \omega + 2$?*

The first unknown instances of this occur at the first gap-ending ordinal $\omega_1^{\text{ck}} + \omega$ and its successor $\omega_1^{\text{ck}} + \omega + 1$. Thus, we don't know whether

$$P_{\omega_1^{\text{ck}}+\omega} = NP_{\omega_1^{\text{ck}}+\omega} \cap \text{co-}NP_{\omega_1^{\text{ck}}+\omega},$$

nor do we know whether

$$P_{\omega_1^{\text{ck}}+\omega+1} = NP_{\omega_1^{\text{ck}}+\omega+1} \cap \text{co-}NP_{\omega_1^{\text{ck}}+\omega+1}.$$

A related question concerns the gap-starting ordinals and their successors, such as ω_1^{ck} and $\omega_1^{\text{ck}} + 1$, where we have proved the equalities

$$P_{\omega_1^{\text{ck}}} = NP_{\omega_1^{\text{ck}}} \cap \text{co-}NP_{\omega_1^{\text{ck}}} \qquad \text{and} \qquad P_{\omega_1^{\text{ck}}+1} = NP_{\omega_1^{\text{ck}}+1} \cap \text{co-}NP_{\omega_1^{\text{ck}}+1}.$$

We will now show that this phenomenon is completely general, appealing to the following unpublished results of Philip Welch.

Lemma 15 ([Wel, Lemma 2.5]) *If α is a clockable ordinal, then every ordinal up to the next admissible beyond α is writable in time $\alpha + \omega$.*

Theorem 16 ([Wel, Theorem 1.8]) *Every ordinal beginning a gap in the clockable ordinals is admissible.*

This latter result is a converse of sorts to [HL00, Theorem 8.8], which establishes that no admissible ordinal is clockable. It is not the case, however, that the gap-starting ordinals are exactly the admissible ordinals below λ, because admissible ordinals can appear in the middle of a gap. To see that this phenomenon occurs, observe that the suprema of the writable and eventually writable ordinals are both admissible, with no clockable ordinals in between, and this situation reflects downwards into an actual gap, because an algorithm can search for accidentally writable admissible ordinals having no clockable ordinals in between, and halt when they are found.

Theorem 17 *Suppose that β begins a gap in the clockable ordinals. Then $P_\beta = NP_\beta \cap \text{co-}NP_\beta$. Furthermore, if β is in addition not a limit of non-clockable ordinals, then $P_\beta = P_{\beta+1} = NP_{\beta+1} \cap \text{co-}NP_{\beta+1}$.*

Proof: Let us suppose first that β begins a gap in the clockable ordinals, but is not a limit of non-clockable ordinals, so that there is some $\nu < \beta$ such that $[\nu, \beta)$ is a contiguous block of clockable ordinals. Since ν is clockable, it follows by Lemma 15 that there is a real u coding ν that is writable in time $\nu + \omega$, which is of course still less than β. We claim that $\beta = \omega_1^u$. To see this, observe that since β is admissible, L_β has the computation producing u and so $u \in L_\beta$. Consequently, β is u-admissible and so $\omega_1^u \leq \beta$. Conversely, since u codes ν and there are no admissible ordinals in $[\nu, \beta)$, it follows that $\beta \leq \omega_1^u$, and so $\beta = \omega_1^u$.

Next, we relativize Theorem 10 with respect to an oracle for u, concluding that $P^u_{\omega_1^u} = NP^u_{\omega_1^u} \cap \text{co-}NP^u_{\omega_1^u} = \Delta^1_1(u) = P^u_{\omega_1^u+1} = NP^u_{\omega_1^u+1} \cap \text{co-}NP^u_{\omega_1^u+1}$, where the superscript indicates the presence of an oracle for u. But since u is writable in time $\nu + \omega < \beta$ by Lemma 15, we can simulate such an oracle by simply taking the time first to write it out. By admissibility, $\nu + \omega + \beta = \beta$, and so this preparatory step will not cause any ultimate delay in our calculations. Therefore, $P_\beta = P^u_\beta$, $NP_\beta = NP^u_\beta$ and $\text{co-}NP_\beta = \text{co-}NP^u_\beta$, and the same for $\beta + 1$. We conclude that $P_\beta = NP_\beta \cap \text{co-}NP_\beta = \Delta^1_1(u) = P_{\beta+1} = NP_{\beta+1} \cap \text{co-}NP_{\beta+1}$, as desired.

It remains to consider the case of gap-starting ordinals β that are limits of gaps. In this case, β is a limit of ordinals ξ that begin gaps but are not limits of non-clockable ordinals (they begin the "successor" gaps), and consequently by the previous paragraph satisfy $P_\xi = NP_\xi \cap \text{co-}NP_\xi = P_{\xi+1} = NP_{\xi+1} \cap \text{co-}NP_{\xi+1}$. Because P_β is the union of the nondecreasing sequence of

classes P_ξ for $\xi < \beta$, and the same for NP_β and co-NP_β, it follows that

$$P_\beta = \bigcup_{\xi<\beta} P_\xi = \bigcup_{\xi<\beta} NP_\xi \cap \text{co-}NP_\xi = (\bigcup_{\xi<\beta} NP_\xi) \cap (\bigcup_{\xi<\beta} \text{co-}NP_\xi) = NP_\beta \cap \text{co-}NP_\beta,$$

and so the proof is complete. □

Corollary 18 *In particular, $P_\lambda = NP_\lambda \cap \text{co-}NP_\lambda$, where λ is the supremum of the clockable ordinals.*

More generally, we ask for a characterization of these exceptional ordinals.

Question 19 *Exactly which ordinals α satisfy $P_\alpha = NP_\alpha \cap \text{co-}NP_\alpha$?*

Just for the record, let us settle the question for the classes P_ω and $P_{\omega+1}$, as well as P_n for finite n, which are all trivial in the sense that they involve only finite computations. The class P_ω concerns the uniformly finite computations, while $P_{\omega+1}$ allows arbitrarily long but finite computations. The class P_n for finite n concerns computations having at most $n-2$ steps. Observe that $P_0 = P_1 = \emptyset$ because computations have nonnegative length, and $P_2 = \{\mathbb{R}, \emptyset\}$ because a computation halts in 0 steps only when the *start* state is identical with either the *accept* or *reject* states. Infinite computations first appear with the class $P_{\omega+2}$.

Theorem 20 *For the classes corresponding to finite computations:*

(*i*) $P_n = NP_n = \text{co-}NP_n$ *for any finite* n. *Consequently,* $P_n = NP_n \cap \text{co-}NP_n$.

(*ii*) $P_\omega = NP_\omega = \text{co-}NP_\omega$. *Consequently,* $P_\omega = NP_\omega \cap \text{co-}NP_\omega$.

(*iii*) $P_{\omega+1} = \Delta^0_1$, $NP_{\omega+1} = \Sigma^0_1$ *and* $\text{co-}NP_{\omega+1} = \Pi^0_1$. *Consequently,* $P_{\omega+1} = NP_{\omega+1} \cap \text{co-}NP_{\omega+1}$.

Proof: For (i), observe that the computations putting a set in P_n, NP_n or co-NP_n are allowed at most $n-2$ many steps, and so the sets they decide must depend on at most the first $n-2$ digits of the input. But any such set is in P_n, because if membership in $A \subseteq \mathbb{R}$ depends on the first $n-2$ digits of the input, then there is a program which simply reads those digits, remembering them with states, and moves to the *accept* or *reject* states accordingly. So

$P_n = NP_n =$ co-NP_n. Claim (ii) follows, because $P_\omega = \bigcup_n P_n = \bigcup_n NP_n = NP_\omega$.

For (iii), observe that a set B is in $P_{\omega+1}$ if $x \in B$ can be decided by a Turing machine program that halts in finitely many steps. Since this is precisely the classical notion of (finite time) computability, it follows that $P_{\omega+1} = \Delta^0_1$, the recursive sets of reals. If $B \in NP_{\omega+1}$, there is an algorithm p such that $\varphi_p(x, y)$ halts in finitely many steps on all input and $x \in B$ if and only if there is a y such that φ_p accepts (x, y). Thus, $x \in B$ if and only if there is a finite piece $y \restriction n$ such that φ_p accepts $(x, y \restriction n)$, where the piece is long enough that the algorithm never inspects y beyond n bits. Since this has now become an existential quantifier over the integers, we conclude that $B \in \Sigma^0_1$. Conversely, every set in Σ^0_1 is clearly the projection of a set in Δ^0_1, so we conclude $NP_{\omega+1} = \Sigma^0_1$. By taking complements, co-$NP_{\omega+1} = \Pi^0_1$. $\Box$

Returning our focus to the infinite computations, let us now consider the case of ordinals that are not necessarily clockable. Our first observation is that the key idea of the proof of Theorem 1—the fact that one could easily recognize codes for ω^ω or any other recursive ordinal—generalizes to the situation where one has only nondeterministic algorithms for recognizing the ordinals in question.

Definition 21 An ordinal α is *recognizable* (in time ξ) when there is a nonempty set of reals coding α that is decidable (in time ξ). The ordinal α is *nondeterministically recognizable* (in time ξ) if there is a nonempty set of codes for α that is nondeterministically decidable (in time ξ).

If α is nondeterministically recognizable in time ξ, then the set WO_α of *all* reals coding α is nondeterministically decidable in time ξ, because a real is in WO_α if and only if there is an isomorphism from the relation it codes to the relation coded by any other real coding α.

Lemma 22 *If an ordinal α is nondeterministically recognizable in time ξ, then $h_\alpha \in NP_{\leq\xi} \cap$ co-$NP_{\leq\xi}$.*

Proof: Suppose that α is nondeterministically recognizable in time ξ, so there is a nonempty set D of codes for α that is in $NP_{\leq\xi}$. We may assume both α and ξ are at least ω_1^{ck}. Consider the algorithm that on input (p, u, v, w) checks, first, that u codes a linearly ordered relation on ω with respect to which v codes the snapshot sequence of $\varphi_p(p)$, showing it to halt,

and second, that (u, w) is accepted by the nondeterministic algorithm deciding D, verifying $u \in D$. If $p \in h_\alpha$, then the computation $\varphi_p(p)$ halts in fewer than α many steps, and so we may choose a real $u \in D$ coding α, along with a real w witnessing that $u \in D$, and a real v coding the halting snapshot sequence of $\varphi_p(p)$, so that (p, u, v, w) is accepted by our algorithm. Conversely, if (p, u, v, w) is accepted by our algorithm, then because (u, w) was accepted by the algorithm for D, we know u really codes α, and so the snapshot sequence must be correct in showing $\varphi_p(p)$ to halt before α, so $p \in h_\alpha$. Finally, the algorithm takes ξ steps, because the initial check takes fewer than ω^2 steps, being arithmetic, and so the computation takes $\omega^2 + \xi = \xi$ many steps altogether. Thus, $h_\alpha \in NP_\xi$.

To see that $h_\alpha \in$ co-NP_ξ, simply modify the algorithm to check that v codes a snapshot sequence with respect to the relation coded by u, but v shows the computation *not* to halt. $\square$

One can use the same idea to show that if NP_α contains a set of codes for ordinals unbounded in α, then $P_\alpha \neq NP_\alpha$.

We will now apply this result to show that $P_\alpha \neq NP_\alpha \cap$ co-NP_α for all sufficiently large countable ordinals α. Recall from the introduction that $\lambda < \zeta < \Sigma$ refer to the suprema of the writable, eventually writable and accidentally writable ordinals, respectively. The first two of these are admissible, while the latter is not, and every computation either halts before λ or repeats the ζ configuration at Σ. And furthermore, Σ is characterized by being the first repeat point of the universal computation simulating all $\varphi_p(0)$ simultaneously.

Theorem 23 *If $\Sigma + 2 \leq \alpha$, then $P_\alpha \neq NP_\alpha \cap$ co-NP_α. In fact, the class $NP_{\leq\Sigma} \cap$ co-$NP_{\leq\Sigma}$ contains a nondecidable set, the halting problem h.*

Proof: The proof relies on the following.

Lemma 23.1 *Σ is nondeterministically recognizable in time Σ.*

Proof: The model-checking algorithm of [HW03, Theorem 1.7] essentially shows this, but let us sketch the details here. By results in [Wel00], the ordinal Σ is the first stage at which the universal computation (simulating $\varphi_p(0)$ for all programs p) repeats itself. Consider the algorithm which on input (x, y) checks whether x codes a relation on ω and y codes a model $M_y \models$ "$KP + \Sigma$ exists" containing x and satisfying the assertion that the

order type of x is Σ^{M_y}. If y passes this test, then the algorithm counts-through the relation coded by x to verify that it is well-founded. If all these tests are passed, then the algorithm accepts in the input, and otherwise rejects it. If the well-founded part of M_y exceeds the true Σ, then M_y will have the correct value for Σ, and the algorithm will take exactly Σ many steps. If the well-founded part of M_y lies below Σ, then this will be discovered before Σ and the algorithm will halt before Σ. Finally, because Σ is not admissible, the well-founded part of M_y cannot be exactly Σ, and so in every case our algorithm will halt in at most Σ many steps. And since the acceptable x have order type Σ, this shows that WO_Σ, the set of reals coding Σ, is nondeterministically decidable in Σ steps, as desired. $\square$

By Lemma 22, it follows that $h_\Sigma \in NP_{\leq\Sigma} \cap \text{co-}NP_{\leq\Sigma}$. But since Σ is larger than every clockable ordinal, it follows that $h_\Sigma = h$, the full lightface halting problem. So we have established that if $\Sigma + 2 \leq \alpha$, then the halting problem h is in $NP_\alpha \cap \text{co-}NP_\alpha$. Since h is not decidable, it cannot be in P_α. So $P_\alpha \neq NP_\alpha \cap \text{co-}NP_\alpha$. $\square$

This establishes $P_\alpha \neq NP_\alpha \cap \text{co-}NP_\alpha$ for all but countably many α. We close this section with another definition and an application.

Definition 24 An ordinal α is *nondeterministically clockable* if there is an algorithm p which halts in time at most α for all input and in time exactly α for some input. More generally, α is *nondeterministically clockable before* β if there is an algorithm that halts before β on all input and in time exactly α for some input.

Such an algorithm can be used as a clock for α in nondeterministic computations, since there are verifying witnesses making the clock run for exactly the right amount of time, with a guarantee that no other witnesses will make the clock run on too long.

Theorem 25 *If α is an infinite nondeterministically clockable limit ordinal, then $P_{\leq\alpha} \neq NP_{\leq\alpha}$.*

Proof: By Lemma 8, it follows that $h_{\alpha+\omega} \notin P_{\leq\alpha}$. But we claim that $h_{\alpha+\omega} \in NP_{\leq\alpha}$. By Theorem 2 we may assume $\alpha > \omega_1^{\mathrm{ck}}$, because when α is recursive $h_{\alpha+\omega}$ is hyperarithmetic, and hence already in $NP_{\omega+2}$. Fix a nondeterministic clock for α, a program e such that $\varphi_e(z)$ halts in exactly α

many steps for some z and in at most α many steps on all other input. We will now nondeterministically decide $h_{\alpha+\omega}$ by the following algorithm. On input (x, y, z), first determine whether x is some finite p. If not, then reject the input, otherwise, check whether y codes a model M_y of KP containing z and satisfying the assertion that $\varphi_e(z)$ halts, with $\varphi_p(p)$ halting at most finitely many steps later. Since this is an arithmetic condition on y, it can be checked in fewer than ω^2 many steps. Next, assuming that these tests have been passed successfully, we verify that the model M_y is well-founded up to what it thinks is the halting time of $\varphi_e(z)$, which we denote α^{M_y}. If ill-foundedness is discovered, we reject the input. By flashing a master flag every time we delete what is the current smallest (in the natural ordering of ω) element still in the field, we can tell at a limit stage that we have finished counting, and when this occurs, we accept the input.

Let's argue that this algorithm accomplishes what we want. First of all, if $p \in h_{\alpha+\omega}$, then $\varphi_p(p)$ halts before $\alpha + \omega$ and there is a real z such that $\varphi_e(z)$ halts in α steps and a real y coding a fully well-founded model $M_y \models KP$ in which these computations exist. So the previous algorithm will accept the input (p, y, z). Conversely, if the algorithm accepts (p, y, z) for some y and z, then the corresponding model M_y is well-founded up to the length of the computation $\varphi_e(z)$, which is at most α because the computation $\varphi_e(z)$ in M_y agrees with the actual computation as long as the model remains well-founded. It follows that model is also well-founded for an additional ω many steps, and so the model is correct about $\varphi_p(p)$ halting before $\alpha + \omega$. So the algorithm does nondeterministically decide $h_{\alpha+\omega}$.

It remains to see that the algorithm halts in at most α many steps on all input. Since $\omega_1^{\mathrm{ck}} \leq \alpha$, it follows that $\omega^2 + \alpha = \alpha$, and so the initial checks of those arithmetic properties do not ultimately cause any delay. The only question is how many steps it takes to check the well-foundedness of M_y up to α^{M_y}. If M_y is well-founded up to α^{M_y}, then this takes exactly α^{M_y} many steps (as the count-through algorithm is designed precisely to take β steps to count through a relation of limit order type β), and this is at most α. If M_y is ill-founded below α^{M_y}, then this will be discovered exactly ω many steps beyond the well-founded part of M_y, and so the algorithm will halt in at most α many steps. Lastly, the well-founded part of M_y cannot be exactly α, because α is not z-admissible. So in any case, on any input the algorithm halts in at most α many steps. $\square$

This argument does not seem to establish that $P_{\leq\alpha} \neq NP_{\leq\alpha} \cap \text{co-}NP_{\leq\alpha}$ for such α, however, because one cannot seem to use a nondeterministic clock in this algorithm to verify that a computation $\varphi_p(p)$ has *not* halted. The problem is that a prematurely halting nondeterministic clock might cause the algorithm to think that $\varphi_p(p)$ does not halt in time $\alpha + \omega$ even when it does, which would lead to false acceptances for the complement of $h_{\alpha+\omega}$.

6 The Cases of P^f and P^{++}

Let us turn now to the question of whether $P^f = NP^f \cap \text{co-}NP^f$, where $f : \mathbb{R} \to \textsc{ord}$. A special case of this is the question of whether $P^{++} = NP^{++} \cap \text{co-}NP^{++}$, because $P^{++} = P^{f_1}$, where $f(x) = \omega_1^x + \omega + 1$. We consider only functions f that are *suitable*, meaning that $f(x) \leq_T f(y)$ whenever $x \leq_T y$ and $f(x) \geq \omega + 1$.

Many of the instances of the question whether $P^f = NP^f \cap \text{co-}NP^f$ are actually solved by a close inspection of the arguments of [HW03], though the results there were stated only as $P^f \neq NP^f$. The point is that the model-checking technique of verification used in those arguments is able to verify both positive and negative answers.

But more than this, the next theorem shows that the analysis of whether $P^f = NP^f \cap \text{co-}NP^f$, at least for sets of natural numbers, reduces to the question of whether $P_\alpha = NP_\alpha \cap \text{co-}NP_\alpha$, where $\alpha = f(0) + 1$. And since the previous section provides answers to this latter question for many values of α, we will be able to provide answers to the former question as well, in Corollaries 27 and 28.

Theorem 26 *For any suitable function f and any set A of natural numbers,*

(i) $A \in P^f$ *if and only if* $A \in P_{f(0)+1}$;

(ii) $A \in NP^f$ *if and only if* $A \in NP_{f(0)+1}$;

(iii) $A \in \text{co-}NP^f$ *if and only if* $A \in \text{co-}NP_{f(0)+1}$.

Proof: By suitability, $f(0) \leq f(x)$ for all x, and $f(0) = f(n)$ for all natural numbers n. Since any set in $P_{f(0)+1}$ is decided by an algorithm that takes fewer than $f(0)$ many steps, it follows that $P_{f(0)+1} \subseteq P^f$. Conversely, suppose that $A \subseteq \omega$ and $A \in P^f$. So there is an algorithm that decides whether $x \in A$

in fewer than $f(x)$ many steps. Although this algorithm might be allowed to take many steps on a complicated input x for which $f(x)$ may be large, we know since $A \subseteq \omega$ that the ultimate answer will be negative unless $x \in \omega$. Thus, we design a more efficient algorithm by rejecting any input x that does not code a natural number. Since the natural number n is coded by the sequence consisting of a block of n ones, followed by zeros, the sequences that don't code natural numbers are precisely the sequence of all ones, plus those containing the substring 01. While continuing with the algorithm to decide A, our modified algorithm searches for the substring 01 in the input, and also turns on a flag if 0 is encountered in the input. This algorithm decides $n \in A$ in fewer than $f(n) = f(0)$ many steps, and rejects all other input either in finitely many steps, if the input contains 01, or in ω many steps, if the input has no zeros. It therefore places A in $P_{f(0)+1}$, as desired.

A similar argument establishes the result for NP^f and $NP_{f(0)+1}$. Specifically, if $A \in NP^f$, then there is a nondeterministic algorithm such that $x \in A$ if and only if the algorithm accepts (x, y) for some y. Once again, we can modify this algorithm to reject any input (x, y) in finitely many steps unless x codes some finite n, in which case the algorithm is carried out as before. The result is that $x \in A$ is decided in finite time unless $x = n \in \omega$, in which case it is decided in fewer than $f(n) = f(0)$ many steps, placing A in $NP_{f(0)+1}$. The result for co-NP^f and co-$NP_{f(0)+1}$ follows by taking complements. $\square$

The argument of [HW03, Theorem 3.1] essentially proves the following result, though that result is stated merely as $P^f \neq NP^f$. Here, we will derive it as a corollary to the previous theorem and Theorem 23. Note that if $f : \mathbb{R} \to \text{ORD}$ is suitable, then $f(q) = f(0)$ for any finite q.

Corollary 27 *If $f : \mathbb{R} \to \text{ORD}$ is suitable and $f(0) > \Sigma$, then P^f is properly contained in $NP^f \cap$ co-NP^f.*

Proof: This follows immediately from Theorems 23 and 26, because the halting problem h, being a set of natural numbers and in $NP_{\leq\Sigma} \cap$ co-$NP_{\leq\Sigma}$, must be in $NP^f \cap$ co-NP^f, but it is not decidable and consequently not in P^f. $\square$

Corollary 28 *If $f : \mathbb{R} \to \text{ORD}$ is suitable and $f(0)$ is clockable, but does not end a gap in the clockable ordinals, then P^f is properly contained in $NP^f \cap$ co-NP^f.*

Proof: By Theorem 26, the sets of natural numbers in P^f and $NP^f \cap \text{co-}NP^f$ are exactly those in $P_{\alpha+1}$ and $NP_{\alpha+1} \cap \text{co-}NP_{\alpha+1}$, respectively, where $\alpha = f(0)$. Since $f(0)$ does not end a gap in the clockable ordinals, it follows that $\alpha + 1$ is neither a gap-ending ordinal nor the successor of a gap-ending ordinal. Therefore, by Corollary 12 there are sets of natural numbers in $NP_{\alpha+1} \cap \text{co-}NP_{\alpha+1}$ that are not in $P_{\alpha+1}$. Consequently, there are sets of natural numbers in $NP^f \cap \text{co-}NP^f$ that are not in P^f. $\square$

An instance of this settles the question for P^{++}.

Corollary 29 *$P^{++} \neq NP^{++} \cap \text{co-}NP^{++}$.*

Proof: This follows from Corollary 28 and the fact that $P^{++} = P^{f_1}$, where $f_1(x) = \omega_1^x + \omega + 1$. By [HL00, Theorem 3.2], the ordinal $\omega_1^{\text{ck}} + \omega$ is clockable, and consequently so is $\omega_1^{\text{ck}} + \omega + 1$. $\square$

So the previous corollaries establish that $P^f \neq NP^f \cap \text{co-}NP^f$ for many or most functions f. But of course, we have examples of ordinals α for which $P_\alpha = NP_\alpha \cap \text{co-}NP_\alpha$, such as $\alpha = \omega_1^{\text{ck}}$ or $\alpha = \omega_1^{\text{ck}} + 1$. If f is the constant function $f(x) = \omega_1^{\text{ck}}$, then it is easy to see that $P^f = P_{\omega_1^{\text{ck}}+1}$ and $NP^f = NP_{\omega_1^{\text{ck}}+1}$, and this provides an example where $P^f = NP^f \cap \text{co-}NP^f$, even when $P^f \neq NP^f$. The equation $P^+ = NP^+ \cap \text{co-}NP^+$ provides another such example.

Vinay Deolalikar
Hewlett-Packard Research
1501 Page Mill Road, M/S 3U-4, Palo Alto, CA 94304
Tel: (650) 857 8605, Fax: (650) 852 3791
http://www.hpl.hp.com/personal/Vinay_Deolalikar/

Joel David Hamkins
Georgia State University & The City University of New York
The College of Staten Island of CUNY and The CUNY Graduate Center
Mathematics Program, 365 Fifth Avenue, New York, NY 10016
http://jdh.hamkins.org

Ralf-Dieter Schindler
Institut für formale Logik, Universität Wien

1090 Wien, Austria
rds@logic.univie.ac.at
http://www.logic.univie.ac.at/~rds/

References

[Bar75] J. Barwise. *Admissible Sets and Structures*. Perspectives in Mathematical Logic. Springer Verlag, 1975.

[HL00] Joel David Hamkins and Andy Lewis. Infinite time Turing machines. *J. Symbolic Logic*, 65(2):567–604, 2000.

[HW03] Joel David Hamkins and Philip Welch. $P^f \neq NP^f$ for almost all f. *Mathematical Logic Quarterly*, 49(5):536–540, 2003.

[Kec95] A. Kechris. *Classical Descriptive Set Theory*. Graduate Texts in Mathematics. Spring-Verlag, New York, 1995.

[Mos80] Yiannis Nicholas Moschovakis. *Descriptive Set Theory*, volume 100 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Company, Amsterdam, New York, Oxford, 1980.

[MW85] R. Mansfield and G. Weitkamp. *Recursive aspects of descriptive set theory*, volume 11 of *Oxford Logic Guides*. Oxford University Press, New York, 1985.

[Sch] Ralf-Dieter Schindler. P $\neq$ NP for infinite time Turing machines. *to appear in Monatshefte für Mathematik*.

[Wel] Philip Welch. The action of one tape machines. July 2, 2003 preprint.

[Wel00] Philip Welch. The lengths of infinite time Turing machine computations. *Bulletin of the London Mathematical Society*, 32(2):129–136, 2000.